HTML5

Modern web applications

Michael Lappenbusch
IT-SPECIALIST APPLICATION DEVELOPMENT

Table of contents

1.Introduction to HTML5

What is HTML5?

HTML5 is the fifth and current version of Hypertext Markup Language (HTML) used to create web pages. It was approved by the World Wide Web Consortium (W3C) in 2014 and has since replaced previous versions such as HTML4 and XHTML.

HTML5 brings many new and improved features compared to previous versions of HTML. Some of the most important innovations are:

New Tags: HTML5 introduces many new tags that allow developers to better structure and present content. Examples are the <article>, <section>, <header> and <nav> tags.

Forms: HTML5 improves the possibilities for creating forms. It supports new input types like date, time, email and color picker. In addition, fields can now be validated directly on the client without requiring a query to the server.

Media: HTML5 facilitates the integration of audio and video content on websites. Developers can now embed media directly into HTML pages instead of relying on external plugins like Flash.

Canvas: HTML5 introduces a new technology called canvas that allows 2D graphics and animations to be rendered directly in the browser.

Geolocation: HTML5 makes it possible to determine a user's geographic position and display it on maps.

Offline storage: HTML5 makes it possible to store data on the user's device even when there is no internet connection.

APIs: HTML5 includes many new APIs that make it possible to integrate advanced functionality into web pages, such as WebSockets, Webworkers, WebRTC and WebGL.

Overall, HTML5 enables developers to create more powerful and interactive websites and applications that are better adapted to different devices and browsers. HTML5 is also important for mobile development as it offers the possibilities for creating responsive design and touch events.

Why should you use HTML5?

There are many reasons why developers should use HTML5. Some of the main advantages are:

Compatibility: HTML5 is compatible with most modern web browsers and devices including desktops, laptops, smartphones and tablets. Developers no longer need to worry about browser incompatibility and plugin issues.

Structured Code: HTML5 introduces many new tags that allow content to be better structured and presented. This makes it easier for developers to organize and maintain their websites and applications more easily.

Media integration: HTML5 makes it possible to embed audio and video content directly into web pages without the need for external plugins such as Flash. This facilitates the creation of interactive and multimedia content.

Interactive Features: HTML5 includes many new APIs that make it possible to create advanced features such as animations, games, and interactive applications.

Offline storage: HTML5 makes it possible to store data on the user's device even when there is no internet connection. This enables the creation of web apps that are also available offline.

Geolocation: HTML5 makes it possible to determine a user's geographic position and display it on maps. This enables the creation of applications that react to the user's current position.

Search engine optimization: The use of HTML5 tags improves the structure of the content, which increases the findability of websites and applications in search engines.

Mobile development: HTML5 is important for mobile development as it offers the possibilities for creating responsive design and touch events.

Overall, HTML5 offers developers many opportunities to create more powerful and interactive websites and applications that are better adapted to different devices and browsers and offer additional functionalities.

Differences between HTML5 and earlier versions of HTML

HTML5 differs from previous versions of HTML in a few important ways. Some of the main differences are:

New Tags: HTML5 introduces many new tags that allow developers to better structure and present content. These new tags relate to semantic elements such as <header>, <nav>, <article>, and <section>. These allow developers to break up the content of their page into logical sections and make them easier for search engines to understand.

Forms: HTML5 improves the possibilities for creating forms. It supports new input types such as date, time, email, color picker and search field, and fields can now be validated directly on the client without the need to go back to the server.

Media: HTML5 facilitates the integration of audio and video content on websites. Developers can now embed media directly into HTML pages instead of relying on external plugins like Flash.

Canvas: HTML5 introduces a new technology called canvas that allows 2D graphics and animations to be rendered directly in the browser.

Geolocation: HTML5 makes it possible to determine a user's geographic position and display it on maps.

Offline storage: HTML5 makes it possible to store data on the user's device even when there is no internet connection.

APIs: HTML5 includes many new APIs that make it possible to integrate advanced functionality into web pages, such as WebSockets, Webworkers, WebRTC and WebGL.

Mobile development: HTML5 is important for mobile development as it offers the possibilities for creating responsive design and touch events.

Backwards Compatibility: HTML5 is backwards compatible with previous versions of HTML, which means that it supports most existing HTML documents and can display them without modification.

Compared to previous versions of HTML, HTML5 is a modern and powerful technology that enables developers to create more powerful and interactive websites and applications. Using new tags and APIs, developers can better structure content, integrate media, add interactive features, and determine a user's geographic location. HTML5 also enables the creation of web apps that are also available offline and it is optimized for mobile development. Overall, HTML5 enables developers to adapt their websites and applications to the rapidly changing demands of modern technology.

2.HTML5 Basics

Structure of an HTML5 document

An HTML5 document has a specific structure made up of different parts. The most important parts of an HTML5 document are:

Doctype: The doctype is the first part of an HTML document and tells the browser that it is an HTML5 document. The doctype for HTML5 is <!DOCTYPE html>

<html> tag: The <html> tag is the root node of the document and contains all other elements. It consists of a start and an end tag.

<head> tag: The <head> tag contains meta information about the document, such as the title of the page, used when the page is viewed in a browser tab or in a bookmark list. It also contains the links to CSS and JavaScript files.

<body> tag: The <body> tag contains the actual content of the document, such as text, images, links, and forms.

<header> Tag: The <header> tag contains the page header, which typically includes the page name, a logo, and a navigation element.

<nav> tag: The <nav> tag contains the navigational elements of the page, such as links to other pages or sections within the page.

<main> Tag: The <main> tag contains the primary content of the page, which should be distinguished from other elements such as the header and navigation.

<article> Tag: The <article> tag contains an independent and self-contained piece of content that can be detached from other parts of the page, such as a blog post or news item.

<section> Tag: The <section> tag contains content that is grouped into logical sections, such as a chapter in a book or a section in a news item.

<footer> Tag: The <footer> tag contains the footer of the page, which usually contains copyright information, links to other sites, or contact information.

Each of these tags should be used with a beginning and ending tag (eg</html>) and each tag has its specific function and purpose. This structure makes it easier for developers to easily organize and maintain their websites and applications. By using semantic tags like <header>, <nav>, <article>, and <section>, developers can break up their page's content into logical sections and make them easier for search engines to understand. This also improves accessibility and user experience for users with assistive technologies.

It is important to note that the structure of an HTML5 document is not static and may differ depending on the needs and purpose of the website or application. Developers can use additional semantic tags or custom classes and IDs to customize the structure of their page.

It is also important that the structure of an HTML5 document is kept clean and tidy to improve readability and maintainability and to avoid page rendering issues. Developers should adhere to existing standards and recommendations to create a consistent and accessible structure for their websites and applications.

The new HTML5 tags

HTML5 introduces many new tags that allow developers to better structure and present content. Here are some of the key new HTML5 tags:

<header> tag: The <header> tag contains the header of a page or section. It can contain the name of the page, a logo, a navigation element, or other meta information.

<nav> tag: The <nav> tag contains the navigational elements of a page, such as links to other pages or sections within the page.

<article> Tag: The <article> tag contains an independent and self-contained piece of content that can be detached from other parts of the page, such as a blog post or news story.

<section> Tag: The <section> tag contains content that is grouped into logical sections, such as a chapter in a book or a section in a news item.

<aside> tag: The <aside> tag contains content that is related to the primary content of the page but is not essential to the understanding of the primary content, such as supplemental information, citations, or advertisements.

<figure> tag: The <figure> tag contains an image or other media file and a caption or legend about it.

<figcaption> tag: The <figcaption> tag contains the caption or caption for a <figure> element.

<time> tag: The <time> tag identifies a date or time and allows it to be formatted and interpreted automatically.

<mark> Tag: The <mark> tag marks a section of text as highlighted and highlighted content.

<progress> Tag: The <progress> tag indicates the progress of a task, such as the progress of a download or form submission.

These new HTML5 tags make it easier for developers to better structure their web pages and applications and make content more meaningful to search engines and users with assistive technologies. They also make it possible to improve the readability and maintainability of the code and to optimize the user experience.

Some of these new tags, such as <header>, <nav>, <article>, <section>, <aside>, and <figure>, allow developers to break up their page's content into logical sections and make them easier for search

engines to understand do. Other tags, such as <time>, <mark>, and <progress>, allow for better formatting of content and adding interactive features.

It's important to note that not all of these new HTML5 tags are required or even useful to be used in every project. Developers should carefully consider which tags are best for their projects to maintain a clean and orderly structure and improve accessibility and user experience.

Text formatting with HTML5

HTML5 offers various ways of formatting text on a web page. Here are some of the most important text formatting tags in HTML5:

<p> Tag: The <p> tag stands for "paragraph" and is used to create paragraphs. The text inside the <p> tag is automatically formatted as a paragraph, with a line break before and after the text.

<h1>-<h6> tag: The <h1>-<h6> tags are used to create headings. The <h1> tag creates the largest heading (the most important), while the <h6> tag creates the smallest heading.

<strong> Tag: The <strong> tag is used to emphasize text and give it meaning. The text inside the <strong> tag is usually bold.

<em> Tag: The <em> tag is used to emphasize text and give it meaning. The text inside the <em> tag is usually in italics.

<abbr> Tag: The <abbr> tag is used to denote abbreviations or acronyms. It allows the user to invoke the full form of the word by clicking or hovering over the tagged word.

<blockquote> tag: The <blockquote> tag is used to display quotes. It creates a section of text that stands out from the rest of the page to emphasize the separation of quotes and actual content.

<q> tag: The <q> tag is used to represent short quotations within the text, it can automatically add quotation marks.

<pre> Tag: The <pre> tag is used to present text in a preformatted manner that preserves spaces and line breaks. It is often used to represent code examples.

<code> Tag: The <code> tag is used to represent individual words or short sections of code within text. It is usually presented in a font called a "code font" to distinguish it from regular text.

<sub> Tag and <sup> Tag: The <sub> tag and <sup> tag are used to represent text as subscript (<sub>) or superscript (<sup>). They are often used to represent mathematical or scientific content.

These text formatting tags make it easier for developers to structure their page's content and give it meaning. They can also be used to improve readability and user experience, and to make content easier for search engines to understand. It is important to use the tags correctly to ensure that the content is interpreted and presented correctly.

Insert links and images

In HTML5, developers can insert links and images in a number of ways to enhance the navigation and visuals of a web page.

Insert links: The <a> tag (anchor tag) is used to create links on a web page. The <a> tag has an "href" attribute that contains the URL of the linked document. Example: Example .

Insert images: The <img> tag (image tag) is used to insert images on a web page. The <img> tag has several attributes, including "src" (the image's URL), "alt" (an alternative description of the image for screen readers and other assistive technologies), "width" and "height" (the dimensions of the image).) and "style" (CSS properties of the image). Example: <img src="image.jpg" alt="example image" width="300" height="200">

There are also other ways to include images, such as the <picture> tag, which makes it possible to provide multiple images with different properties to ensure optimal display on different devices.

It is important for developers to ensure that the links and images used are correct and up-to-date in order to ensure smooth navigation and correct presentation of content on the website. It's also important to provide alternative text descriptions for images to improve accessibility for users with assistive technologies.

3.HTML5 forms

Form Elements and Attributes

HTML5 provides various form elements and attributes that enable developers to create interactive forms on a web page. Here are some of the most important form elements and attributes in HTML5:

<form> Tag: The <form> tag defines a form on a web page. Various form elements such as text fields, selection menus and buttons can be placed within the form.

<input> tag: The <input> tag creates various types of form elements, such as text fields, check boxes, radio buttons, and buttons. The "type" attribute determines the type of form element, eg "text" for a text field or "checkbox" for a checkbox.

<label> tag: The <label> tag creates a label for a form element. It allows users to select a form element with the keyboard by clicking on the label.

<select> tag: The <select> tag creates a selection menu. Within the <select> tag, <option> tags can be used to add individual choices.

<textarea> tag: The <textarea> tag creates a text field for multiple lines. It differs from a plain <input> tag with "text" as the type attribute, as it allows multiple lines of text to be entered.

"required" attribute: The "required" attribute can be used to mark a form element as a mandatory field. A form element with this attribute must be filled out before the form can be submitted.

"pattern" attribute: The "pattern" attribute can be used to specify a regular expression that should validate the input in the form element. Example: <input type="text" pattern="[a-zA-Z0-9]+" required>

"placeholder" attribute: The "placeholder" attribute can be used to display preliminary text in the form element that describes what should be entered into the field. Example: <input type="email" placeholder=" your-email@example.com ">

"min" and "max" attributes: These attributes can be used to set the minimum and maximum values for numeric input in the form element. Example: <input type="number" min="1" max="100">

These form elements and attributes make it easy for developers to create interactive forms on a web page and collect data from users. It is important to carefully plan and test the use of these elements and attributes to ensure they work correctly and data security is maintained.

Input Types and Validation

HTML5 provides different input types for the <input> tag, allowing developers to restrict and validate user input to specific types of data. Here are some of the main input types in HTML5:

text: The text input type creates a text field in which users can enter any text.

password: The input type "password" creates a text field in which users can enter a password. The text you type is usually replaced with asterisks or other escape characters.

email: The email input type creates a text field for users to enter an email address. It can automatically verify that the entered email address is in a valid format.

number: The number input type creates a text field in which users can enter numeric values. It can automatically check whether the entered value is within a certain range.

url: The url input type creates a text field in which users can enter a URL. It can automatically verify that the entered URL is in a valid format.

date: The "date" input type creates a text field or date picker field in which users can enter a date. It can automatically verify that the entered date is in a valid format.

search: The "search" input type creates a text field, which is similar to "text", but it is often used to implement a search function.

These input types make it easier for developers to validate input from users and ensure that the data they enter is in the expected format. It is important to note that support for these input types may vary between different browsers and it is recommended to check compatibility.

There are also JavaScript validations that can be added to validate input and display error messages if necessary. These validations can be implemented in different ways, eg using regular expressions, using the HTML5 "pattern" attribute or using JavaScript functions that are applied to the form or individual form elements.

It is important to carefully plan and test input validation to ensure it is working correctly and data security is maintained.

Send and receive form data

In order to send and receive form data from a web page, developers need to follow a few steps:

Creating the form using the <form> tag and various form elements such as text boxes, drop down menus and buttons.

Specifying the action URL to which the form will be submitted using the "action" attribute of the <form> tag. Example: <form action="submit-form.php">

Specifying the transfer method type with the "method" attribute of the <form> tag. There are two main methods: "GET" and "POST". GET is used to send data in URL parameter while POST is used to send data in HTTP body.

Adding a button or other element to the form that causes the form to be submitted, such as <input type="submit" value="Submit">

Receiving and processing the form data on the server side. This can be done using a programming language such as PHP, Ruby, Python or Node.js.

Submit the form with JavaScript or by clicking the Submit button

Checking the data on the server side, e.g. through validations, and sending feedback to the user if necessary.

Saving or further processing of the data in a database or another storage medium.

It is important to ensure that the form data is transmitted safely and securely and that it is processed correctly on the server side. It is also important to ensure that data protection regulations are adhered to.

4.HTML5 Media

Audio and video integration

HTML5 allows developers to embed audio and video content into a web page in a number of ways.

Audio embedding: The <audio> tag makes it possible to embed audio files in a web page. <source> tags can be used within the <audio> tag to indicate the source of the audio file. Example:

```
<audio controls>

<source src="audio.mp3" type="audio/mpeg">

<source src="audio.ogg" type="audio/ogg">

</audio>
```

Video embedding: The <video> tag makes it possible to embed video files in a web page. <source> tags can be used within the <video> tag to indicate the source of the video file. Example:

```
<video controls>

<source src="video.mp4" type="video/mp4">

<source src="video.ogg" type="video/ogg">

</video>
```

The "controls" attribute in audio and video tags allows the user to control the playback of the video or audio (play/pause/volume/etc).

It is important to ensure that the embedded audios and videos are correctly formatted and compatible with most modern browsers. It is also advisable to provide alternative content in case the embedded audio or video is not supported on some devices or browsers.

There are also alternative methods to embed audio and video content, such as using iframe or embed tags from external services like YouTube or Vimeo.

Media control and scaling

HTML5 offers several ways to enable control and scaling of media content such as audio and video on a web page.

Media Controls: The <audio> and <video> tags provide controls such as play/pause, volume control, and progress bar by default. Developers can also use JavaScript to add additional controls such as the ability to jump to a specific time in the audio or video content.

Media scaling: By default, the <audio> and <video> tags provide the ability to adjust the size of the embedded content by defining the width and height of the tag. CSS also allows developers to further customize the appearance of the content, for example by changing the dimensions, background, or borders.

Full screen mode: In HTML5 there is the possibility to display a video or an audio in full screen mode, this can be achieved through JavaScript using the requestFullscreen() method on the video or audio element.

Autoplay: There are some limitations regarding the autoplay of audios and videos on a webpage as they can often be found annoying. Some browsers block autoplay of content unless the user has explicitly allowed it. However, developers can use JavaScript to enable autoplay of audios or videos, however, they should ensure that this feature is carefully planned and implemented to ensure a positive user experience.

It is important that developers carefully plan and test media content controls and scaling to ensure they work correctly and enhance the user experience. It's also important to consider the limitations and best practices for content autoplay to ensure a positive user experience.

Support for different media formats

Support for different media formats varies by browser and platform. Some formats are supported by all modern browsers, while others are only supported by specific browsers. To ensure compatibility, it's a good practice to provide multiple formats and use the <source> tag inside the <audio> or <video> tag to indicate the different formats.

Audio Formats: The most commonly supported audio formats are MP3, WAV, and OGG. MP3 and WAV are the most common, while OGG is supported by some browsers like Firefox and Chrome.

Video Formats: The most commonly supported video formats are MP4, WebM, and OGG. MP4 is the most widely used and supported by most browsers while WebM and OGG are supported by some browsers like Firefox and Chrome.

There are also alternative methods of delivering audio and video content, such as using external services such as YouTube or Vimeo, which host the content in a single format and can be embedded via iframe or embed tags. This can circumvent the compatibility issues, but it can also create dependencies on these services and possible restrictions on the use of the content.

It is important to note that support for certain media formats may change over time and it is a good idea to check compatibility regularly. It is also important to provide alternative formats to ensure that the content can be played on as many devices and browsers as possible.

5.HTML5 Canvas

Drawing and painting with canvas

HTML5 provides the ability to draw and paint using the <canvas> tag. The <canvas> tag creates a transparent rectangular element that developers can use JavaScript to create drawings and graphics on.

Drawing on the canvas: JavaScript allows developers to draw on the canvas by accessing the <canvas> element and using methods such as getContext to access the canvas context object. Then they can use methods like "fillRect" or "strokeRect" to draw rectangles on the canvas. There are also methods like "moveTo" and "lineTo" that can be used to draw lines, and methods like "arc" or "ellipse" to draw circles and ellipses.

Colors and Line Styles: Using JavaScript, developers can set the color and line style for the drawings on the canvas. You can use methods like "fillStyle" or "strokeStyle" to set the color and use methods like "lineWidth" to change the thickness of the lines.

Draw images and other canvas: Developers can also draw images and other canvas elements on a canvas using the "drawImage" method. You can also save Canvas as an image using the "toDataURL" method.

There are also numerous libraries and frameworks that make Canvas easier to use and provide additional functionality such as animations, transformations, and event handling. It is important to ensure that the drawings and graphics on the canvas are carefully planned and tested to ensure a positive user experience.

Animations and Interactions

HTML5 provides the ability to create animations and interactions using JavaScript and CSS.

CSS Animations: CSS allows developers to create animations using the "transition" or "animation" property. You can also create keyframe animations by defining @keyframes rules and applying them to an element. These types of animations are easy to implement, but are limited in terms of the number of properties that can be animated and the control options.

JavaScript animations: JavaScript allows developers to create animations by accessing elements and changing their properties over time. One way is to use timing functions like setInterval or requestAnimationFrame to update the animations. JavaScript animations offer more control and flexibility than CSS animations, but are more expensive to implement.

Interactions: JavaScript allows developers to create interactions by applying event listeners to elements and responding to specific events. Examples of such events are mouse clicks, touches on mobile devices, and keystrokes. Developers can use these events to control the animations, add or remove elements, save data, and more.

There are also numerous libraries and frameworks that make it easy to create animations and interactions, such as jQuery, GreenSock, and Anime.js. It is important to ensure that the animations and interactions are carefully planned and tested to ensure a positive user experience and not impact performance.

Optimize canvas performance

Canvas animations and interactions can perform poorly if drawing and updating the canvas is too frequent or too expensive. There are a few techniques developers can use to optimize canvas application performance:

Using requestAnimationFrame instead of setInterval or setTimeout: requestAnimationFrame synchronizes the canvas refresh with screen refresh rates, resulting in a smoother animation and improving performance.

Use cached variables and objects: Avoid creating unnecessary variables and objects that have to be recreated every frame. Caching these variables and objects can improve performance.

Using transparency: Transparent areas of the canvas don't need to be drawn, which improves performance.

Use compositing: Use the canvas's compositing methods to draw multiple elements in a single step, rather than drawing each element individually.

Avoid unnecessary updates: Avoid unnecessary canvas updates by using cached values and check if an update is really needed.

Avoid unnecessary calculations: Avoid unnecessary calculations in each frame by using cached values and performing calculations in advance.

It's important to regularly monitor and tweak the performance of canvas apps to ensure they run smoothly across devices and browsers. It's also a good idea to use the browser's development tools to monitor and optimize performance.

6.HTML5 Geolocation

Location determination with HTML5

HTML5 allows developers to determine the user's location by using the Geolocation API. This API allows a web application to get access to the user's GPS data and thereby determine the user's current position.

The Geolocation API has a simple JavaScript interface that allows developers to query the user's current location. These include the getCurrentPosition and watchPosition methods, which can query and monitor the user's current position. Both methods accept a callback that is invoked when the user's position is successfully queried.

It is important to note that soliciting the user's location may compromise user privacy and it is a good practice to ask users for location permission before soliciting the position.

There are also alternative methods to enable location determination, such as using IP address or Wi-Fi networks. However, these methods are typically not as accurate as GPS and may not always be reliable.

View maps and placemarks

HTML5 allows developers to display maps and placemarks in web applications using external map APIs like Google Maps or OpenStreetMap. These APIs provide JavaScript libraries that allow developers to embed a map in a web page and add placemarks.

The implementation of a map in a web application usually consists of the following steps:

Obtaining an API key from the chosen map API

Integrating the JavaScript library of the API into the website

Create a <div> element in which to display the map

Initialize the map using the JavaScript library and API key

Adding placemarks to the map using JavaScript library methods

When using placemarks, developers can add various pieces of information, such as the name, address, an image, or a description of the placemark. There are also ways to provide interactions with placemarks such as showing pop-up windows or links to more information.

There are also alternative methods to display maps and placemarks, such as using WebGL-based libraries like Mapbox GL JS or Deck.gl. These libraries make it possible to integrate interactive and high-performance maps into a web application and offer advanced functions such as 3D displays and the ability to display large amounts of data.

It is important to note that the use of Maps APIs usually incurs costs or usage restrictions and it is advisable to read and understand the Terms of Service carefully. It is also important to consider user privacy and ensure that privacy policies are followed.

Security and Privacy

Security and privacy are important considerations when developing web applications, especially when they handle sensitive data such as personal information or payment details.

An important aspect of security is the transmission of data between the user and the application. This should always be done over a secure connection known as "HTTPS". This connection uses an SSL or TLS certificate to ensure that the data is encrypted and cannot be intercepted by third parties.

Another important aspect is the validation of input data. This should always be done on server side to ensure only valid data is processed. It is also important that the application is protected against attacks such as SQL injection or Cross-Site Scripting (XSS).

In terms of privacy, it is important that the application respects the privacy policy and that the user is informed about the use of their data. This should be set out in a privacy statement that should be easily accessible. It should also be ensured that the application only collects the necessary data and does not keep it longer than necessary.

It is important that developers regularly install security updates and patches for the technologies and libraries used to protect their applications against known vulnerabilities. It's also good practice to run regular security tests to identify and fix potential vulnerabilities.

7.HTML5 storage

Local memory

HTML5 allows developers to store data locally in the user's browser without requiring a connection to the server. This is made possible by using the Web Storage API, which provides two types of local storage: "localStorage" and "sessionStorage".

The localStorage is used to store data that should be retained across multiple sessions. This means that the saved data is preserved even if the user closes the application or restarts the browser.

The sessionStorage is used to store data for a single session only. This means that the stored data will be deleted as soon as the user closes the application or restarts the browser.

Both types of storage are key-value based, and developers can store and retrieve data using methods such as setItem() and getItem(). There are also methods for removing data from storage and checking available storage capacity.

It is important to note that local storage is limited and that older browsers may not be fully compatible with the Web Storage API. Developers should therefore ensure that the application works in older browsers and that they store the data safely and securely.

session storage

Session Storage is part of HTML5's Web Storage API that allows developers to store data in the user's browser. In contrast to local storage, which stores data permanently, session storage is intended to store data only for the duration of a single session. A session begins when the user launches the application and ends when the user closes the application or restarts the browser.

Session Storage uses a key-value-based storage model, allowing developers to store and retrieve data using methods such as setItem() and getItem(). There are also methods for removing data from storage and checking available storage capacity.

A use case for session storage can be that a web application wants to save the state of the user during a session without having to save this state permanently. For example, an e-commerce application could store the user's shopping cart in session storage so that the user does not lose their selection after closing or restarting the browser.

It is important to note that session storage is limited and that older browsers may not be fully compatible with the Web Storage API. Developers should therefore ensure that the application works in older browsers and that they store the data safely and securely.

cookies

Cookies are small text files that are stored on the user's computer by a website when the user visits the site. They are used to store information about the user and their interactions with the website in order to improve the user experience and increase the functionality of the website.

Cookies are sent from a web server to the user's browser and stored by it. The next time you visit the website, the server can retrieve the stored cookies and thereby recognize who the user is and what actions he has previously performed on the website.

Cookies can store various types of information, such as the user's username and password to log in automatically, the contents of the shopping cart in an e-commerce store, or the user's preferences, such as language settings.

Cookies can be used in different ways and there are different types of cookies, such as:

Session cookies, which are only stored for the duration of a session and are automatically deleted when the browser is closed

Persistent cookies, which are stored for a certain period of time or indefinitely and are only deleted after this time or manually

First-party cookies, set by the website that the user visited

Third-party cookies, set by a domain other than the website you are visiting, such as advertising partners or analytics tools.

There are also special types of cookies such as secure cookies and HttpOnly cookies that have certain restrictions on access to the cookies to increase security.

It is important to note that cookies may pose privacy issues as they enable tracking of user activity and collection of personal information. Therefore, in many countries there are legal regulations that regulate the use of cookies and require the user's consent. It is therefore important that developers comply with privacy policies and give users the option to accept or refuse the use of cookies.

8.HTML5 APIs

WebSockets

WebSockets are a protocol that makes it possible to set up bidirectional, persistent connections between a web client and a server. Unlike traditional HTTP connections, where each request from a client to a server requires a separate connection, a WebSocket connection allows both the client and server to send and receive data at any time.

WebSockets are established via an upgrade request from HTTP to the WebSocket protocol. Once the connection is established, both the client and server can send and receive messages over the connection without having to establish a new connection.

This protocol enables developers to create applications that can react to changes in real time, such as online gaming, chat applications, financial monitoring, and more. It also enables the creation of applications with reduced network traffic and latencies since the connections remain open and no new connections need to be established.

It is important to note that not all browsers and servers support WebSockets and there may be limitations when using firewalls or other network restrictions. Developers should therefore ensure that their applications work with older browsers and in limited network environments, and that they ensure the security of WebSocket connections, for example by implementing authentication and encryption.

WebSockets requires WebSocket protocol support on both client and server side. There are various libraries and frameworks that make it possible to use WebSockets, both on the client and server side.

There is also a way to use WebSockets via a proxy server or a load balancer, which allows the connections to be shared between multiple servers, thus spreading the load across multiple servers.

Overall, WebSockets enables a new type of interaction between client and server, enabling developers to build applications that can respond to changes in real time and provide greater user interactivity.

webworkers

Web workers are a feature of HTML5 that allow JavaScript code to run in the background without affecting the user interface. They allow long-running processes, such as calculations or data queries, to be run in their own thread, which can increase the performance and responsiveness of the application.

A web worker is started by creating a new worker instance using the Worker() constructor and can then be controlled by calling methods such as postMessage() and terminate(). The web worker has its own global context and therefore cannot directly access the main thread's DOM elements or variables. Instead, data must be exchanged between the main thread and the web worker via the "postMessage()" method.

Web Workers can run multiple at once, allowing developers to parallelize tasks such as data processing, thereby improving application performance. It is important to note that not all browsers support Web Workers and there may be limitations when using certain features or APIs. Developers should therefore ensure that their applications are compatible with older browsers and that they check Web Workers support before using this functionality in their application.

It's important to note that web workers don't have access to certain APIs, such as the DOM or most JavaScript APIs. Web workers also cannot perform GUI actions, they can only calculate data and send the results back to the main thread.

Web workers are useful for tasks that take a long time and don't necessarily affect user interaction. They are particularly useful for tasks such as data processing, image or audio processing, and other tasks that require a lot of computing power.

Overall, using Web Workers allows for improving application performance and responsiveness by allowing long-running processes to run in the background. However, developers should ensure that their applications are compatible with older browsers and that they consider security and privacy when using Web Workers in their application.

WebRTC

WebRTC (Real-Time Communication) is a technology that makes it possible to conduct real-time audio and video communication directly in the browser without the need for additional plugins or software. It's an open standard and supported by many modern browsers including Chrome, Firefox, Safari and Edge.

WebRTC allows developers to build applications that enable audio and video chat, peer-to-peer file transfers, and even virtual and augmented reality. It uses different technologies for this, such as RTP (Real-time Transport Protocol), STUN (Session Traversal Utilities for NAT) or TURN (Traversal Using Relays around NAT)

WebRTC also offers advanced features such as support for multiple audio and video streams, the ability to adjust the size and position of video feeds, and the ability to control audio and video settings. It also allows the use of data channels that allow data to be exchanged in real time between participants.

It is important to note that not all browsers support WebRTC and there may be limitations when using certain functions or APIs. Developers should therefore ensure that their applications are compatible with older browsers and that they verify WebRTC support before implementing this functionality in their applications. Also the privacy and security aspects should be considered when using WebRTC as there may be data leaks or unauthorized access to the transmitted data.

Another important area of application of WebRTC is the connection of external devices such as webcams or microphones to a web application, which represents a simple and cross-platform possibility for using these devices. The possibility of using WebRTC in mobile applications, such as hybrid apps or progressive web apps, further expands the possible uses of WebRTC.

In our book we will go into detail about the different aspects of WebRTC and show how best to integrate it into your own applications. We will also provide examples of typical applications and use cases of WebRTC and discuss the main best practices for developing applications with WebRTC.

WebGL

WebGL (Web Graphics Library) is an extension of HTML5 that makes it possible to display 3D graphics and animations directly in the browser. It is based on the OpenGL API (Application Programming Interface) originally used for developing applications on desktop systems. Using WebGL, developers can deliver interactive 3D content directly into a web page or web application without the need for additional plugins or software.

WebGL makes it possible to display 3D models, textures, lighting and shadows in a web page. It also supports the use of shader programs that allow to customize and optimize the appearance of 3D content. By using WebGL, developers can create interactive 3D applications such as games, simulations, medical visualization, architectural visualization, and more.

Another important area of application of WebGL is its use in virtual and augmented reality. With the ability to display 3D content directly in the browser, VR and AR applications can be developed without the need for additional software or hardware.

It is important to note that not all browsers support WebGL and there may be limitations when using certain functions or APIs. Developers should therefore ensure that their applications are compatible with older browsers and that they verify WebGL support before implementing this functionality in their applications. The performance of WebGL applications can also vary depending on the user's hardware equipment and should be taken into account during development.

In our book we will go into detail about the different aspects of WebGL and show how best to integrate it into your own applications. We will also provide examples of typical applications and use cases of WebGL and discuss the main best practices for developing applications with WebGL. This also includes how to optimize performance to ensure smooth display of 3D content. We'll also cover the different ways one can include 3D models and textures in a WebGL application, as well as the different shader programs available to customize and tweak the rendering.

Another important topic that we will cover in our book is the use of WebGL in virtual and augmented reality. We will address the technical challenges associated with the development of VR and AR applications, as well as the different ways of designing the interaction with 3D content in a VR or AR environment.

Overall, WebGL offers a powerful way to display interactive 3D content directly in the browser and enables developers to develop a variety of applications and use cases. In our book we will go into all aspects of WebGL and show how best to integrate it into your own applications.

9.HTML5 Mobile Development

Responsive design

Responsive design is a concept in web development that aims to ensure that websites and applications are optimally displayed on different devices and screen sizes. It allows the content and layout of a web page to be automatically adjusted to the respective screen size without the user having to manually switch between different versions of the web page.

Responsive design is based on the use of flexible layouts and grids, which allow the elements of a website to be automatically adjusted to the respective screen size. Percentages are usually used for the width and height of the elements instead of fixed pixel values. The use of flexible images and fonts also enables the website content to be optimally displayed on different devices.

An important part of responsive design is the use of media queries. Media queries make it possible to apply certain CSS rules only when certain conditions are met, such as the screen size. For example, certain layout rules can only be applied if the screen size is smaller than a certain value.

Responsive design has gained importance in recent years as more and more users access websites and applications with different devices and screen sizes. It enables better usability and increases user satisfaction because the content is always presented in the best possible way. Responsive design is also of great importance for search engine optimization, as it helps ensure that the content of a website is easily accessible and readable on different devices.

In our book, we will go into detail about the different aspects of responsive design and show how best to integrate it into your own websites and applications. We will discuss the different techniques that can be used to implement responsive design, such as flexible grid system, flexible layout, media queries and the use of flexible images and fonts. We will also provide examples of typical applications and use cases of responsive design and discuss the main best practices for responsive website development.

We will also address the challenges that can arise when implementing responsive design, such as adapting content and layouts to different devices and screen sizes, optimizing performance and avoiding problems with navigation or accessing certain functions .

An important aspect of responsive design is testability on different devices and screen sizes. We will go over the different tools and methods developers can use to test their websites and applications on different devices and identify problems.

All in all, responsive design offers a powerful way to ensure that websites and applications are displayed on different devices and screen sizes. In our book we will go into all aspects of responsive design and show how best to integrate it into your own websites and applications.

touch events

Touch events are events used in web development to capture and process user interaction with touch screens. They allow users to perform interactions such as tapping, swiping, zooming, and other actions on a touch screen.

The most important touch events are touchstart, touchmove, touchend and touchcancel.

touchstart is triggered when the user touches the touchscreen.

touchmove is triggered when the user moves their finger on the touchscreen.

touchend is triggered when the user takes their finger off the touchscreen.

touchcancel is triggered when a touch event is canceled unexpectedly, e.g. by a call or a notification.

Each touch event also returns information about the exact location of the touch point on the touchscreen. This information can be used to control interaction with certain elements on a website or web application.

There are also advanced touch events such as touchforcechange and touchleave, which allow the force and duration of a touch to be captured and processed.

There are also gesture events like pinch and rotate that allow to capture and process the interactions like zoom and rotate.

It is important to note that not all browsers support touch events and there may be limitations when using certain functions or APIs. Developers should therefore ensure that their applications are compatible with older browsers and that they verify touch event support before implementing this functionality in their applications. Some browsers also require specific polyfills or libraries to ensure touch event support.

iOS and Android support

iOS and Android are the two main operating systems for mobile devices and offer different ways of developing applications.

iOS is Apple's operating system and is used on iPhones, iPads and iPods. Applications for iOS are usually developed using the Swift or Objective-C programming language and the Xcode development environment. iOS provides an extensive library of APIs and frameworks for developing applications that run on the various Apple devices. There are also many tools and resources for iOS application development, including design guidelines and best practices from Apple.

Android is an operating system developed by Google and available on a variety of devices from different manufacturers. The development of applications for Android is usually done using the Java or Kotlin programming language and the Android Studio development environment. Android also provides an extensive library of APIs and frameworks for developing applications that run on the various Android devices. There are also many tools and resources for Android application development, including design guidelines and best practices from Google.

It is important to note that iOS and Android support may differ and there may be limitations when using certain functions or APIs. Developers should therefore ensure that their applications are compatible with the latest versions of iOS and Android and that they verify iOS and Android support before implementing this functionality in their applications.

imprint

Author: Michael Lappenbusch

E-mail:admin@perplex.click

home page:https://www.perplex.click

Release year: 2023

Some of the content comes from:ChatGPT